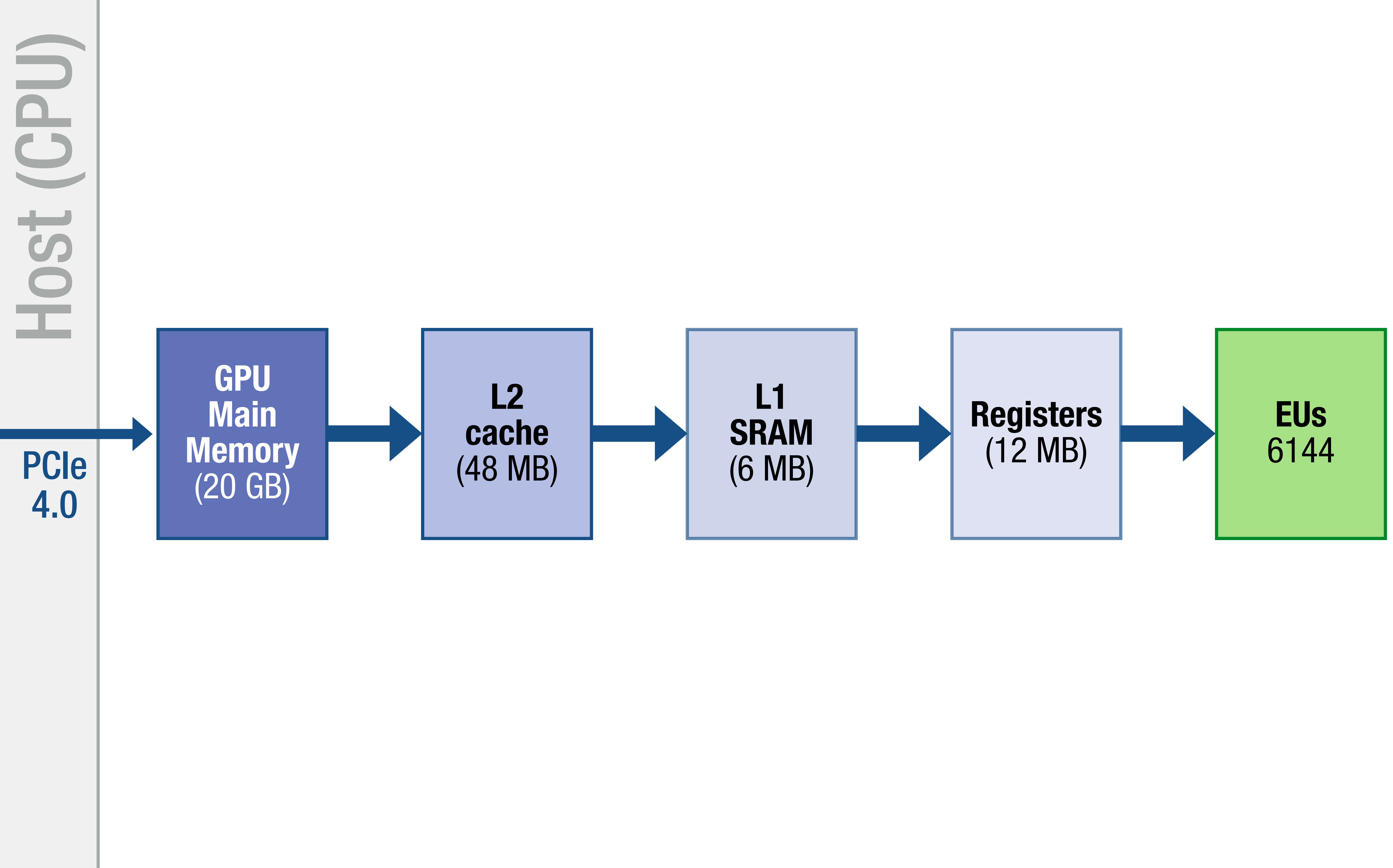
September 25, 2025

# 6.S894 Accelerated Computing

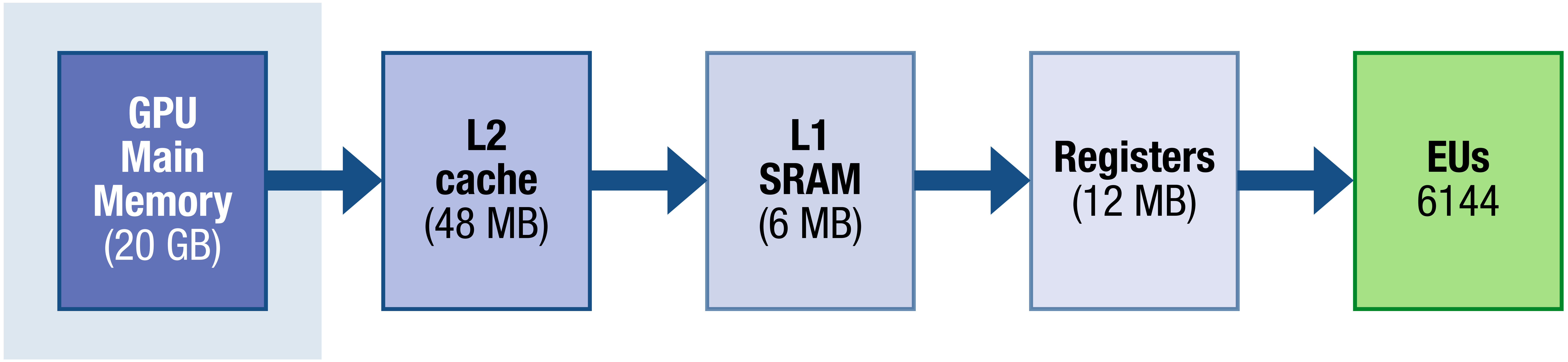
## Lecture 4: Memory
Continued…

Jonathan Ragan-Kelley


Host (CPU)
PCIe 4.0
GPU Main Memory (20 GB)
L2 cache (48 MB)
L1 SRAM (6 MB)
Registers (12 MB)
EUs 6144

1 load /
150 ops

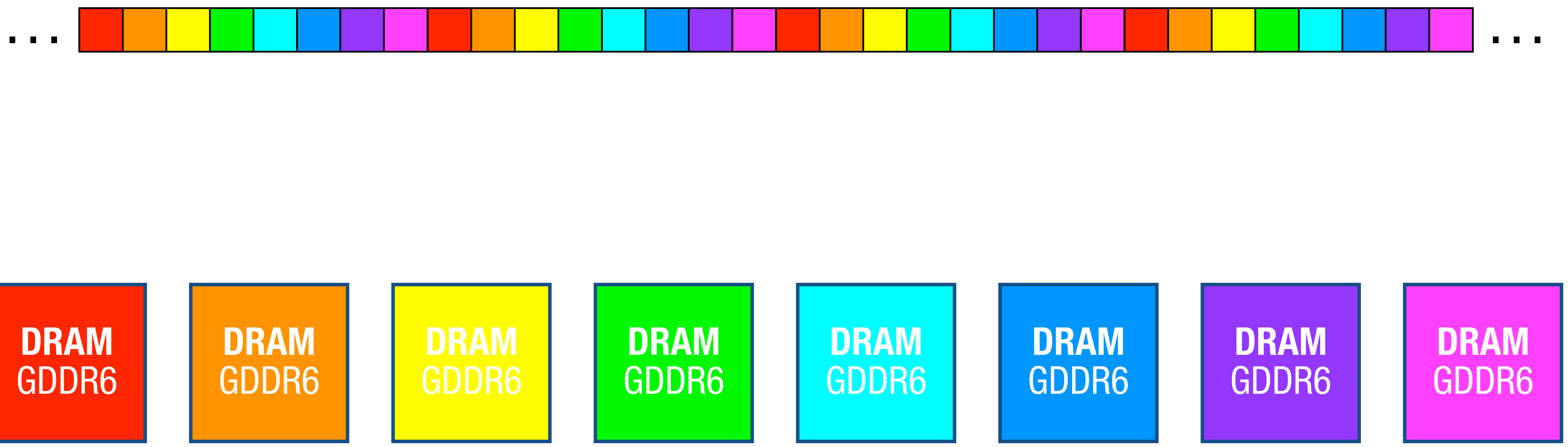
360
GB/sec



high **bandwidth**,
limited **capacity**

high clocks &
wide interface

# Memory is **striped across channels** for high bandwidth on **contiguous access**



32 x 32 bits/cycle
EU EU EU EU EU EU EU EU EU EU EU EU EU EU EU EU EU EU EU EU
32 x 32 bits/cycle

1024 bits/cycle
EU
1024 bits/cycle

**gather**

32 x 32 bits/cycle

...
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
EU
...

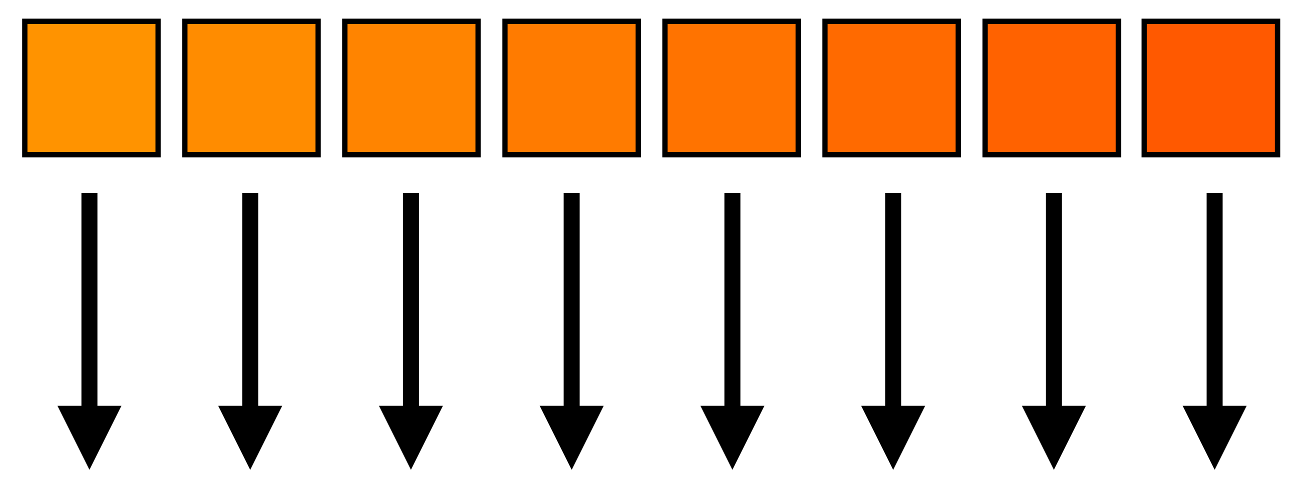
32 x 32 bits/cycle

**scatter**

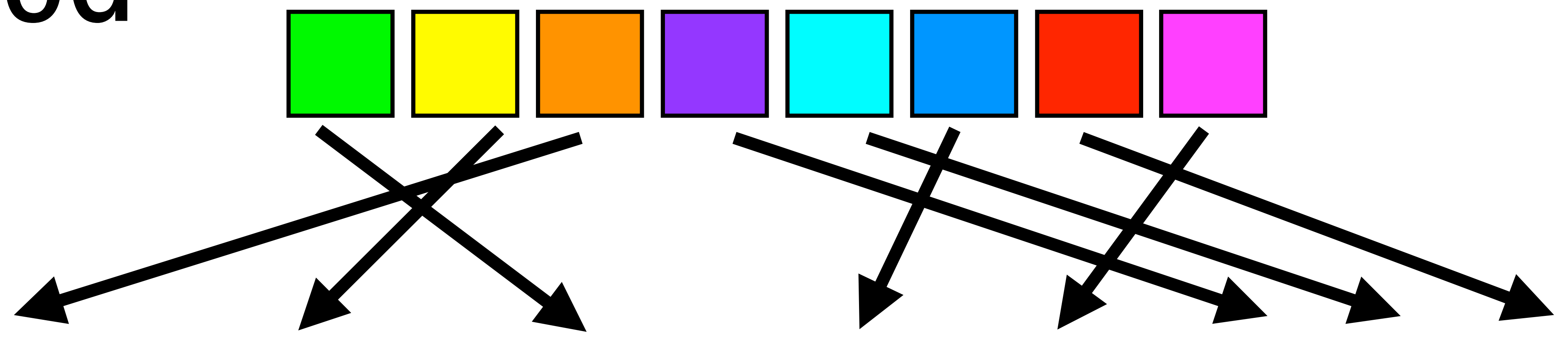
How can we turn **gather/scatter** into **dense load/store** to DRAM?

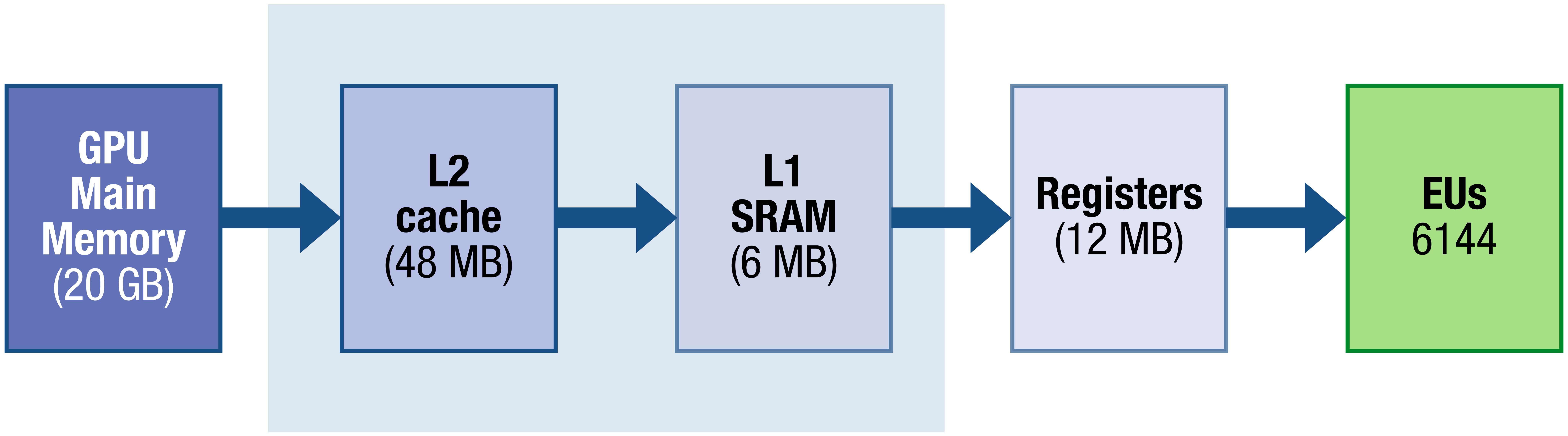
# **Approach 1:** "coalescing" at the memory controller

coalesced access:

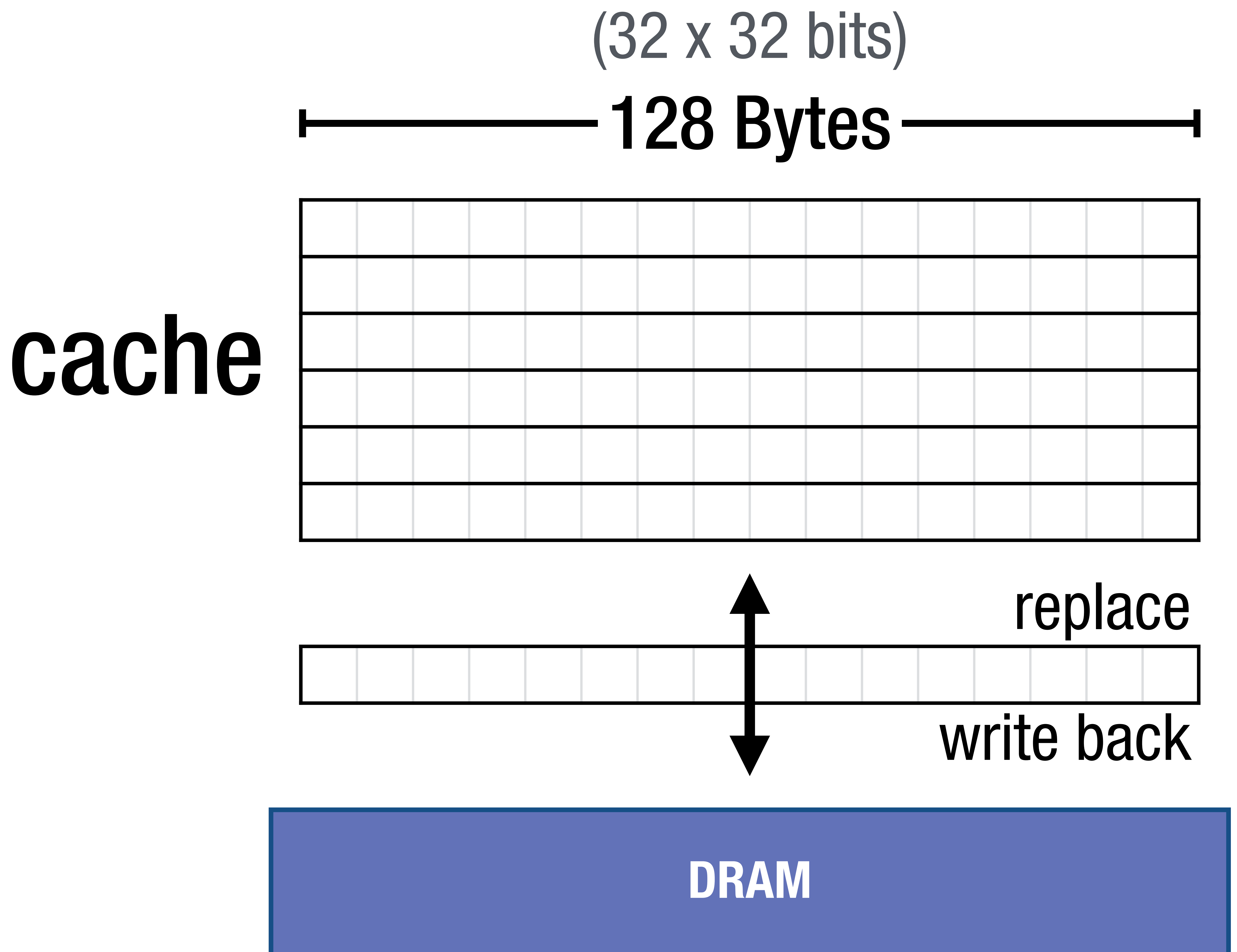


uncoalesced access:

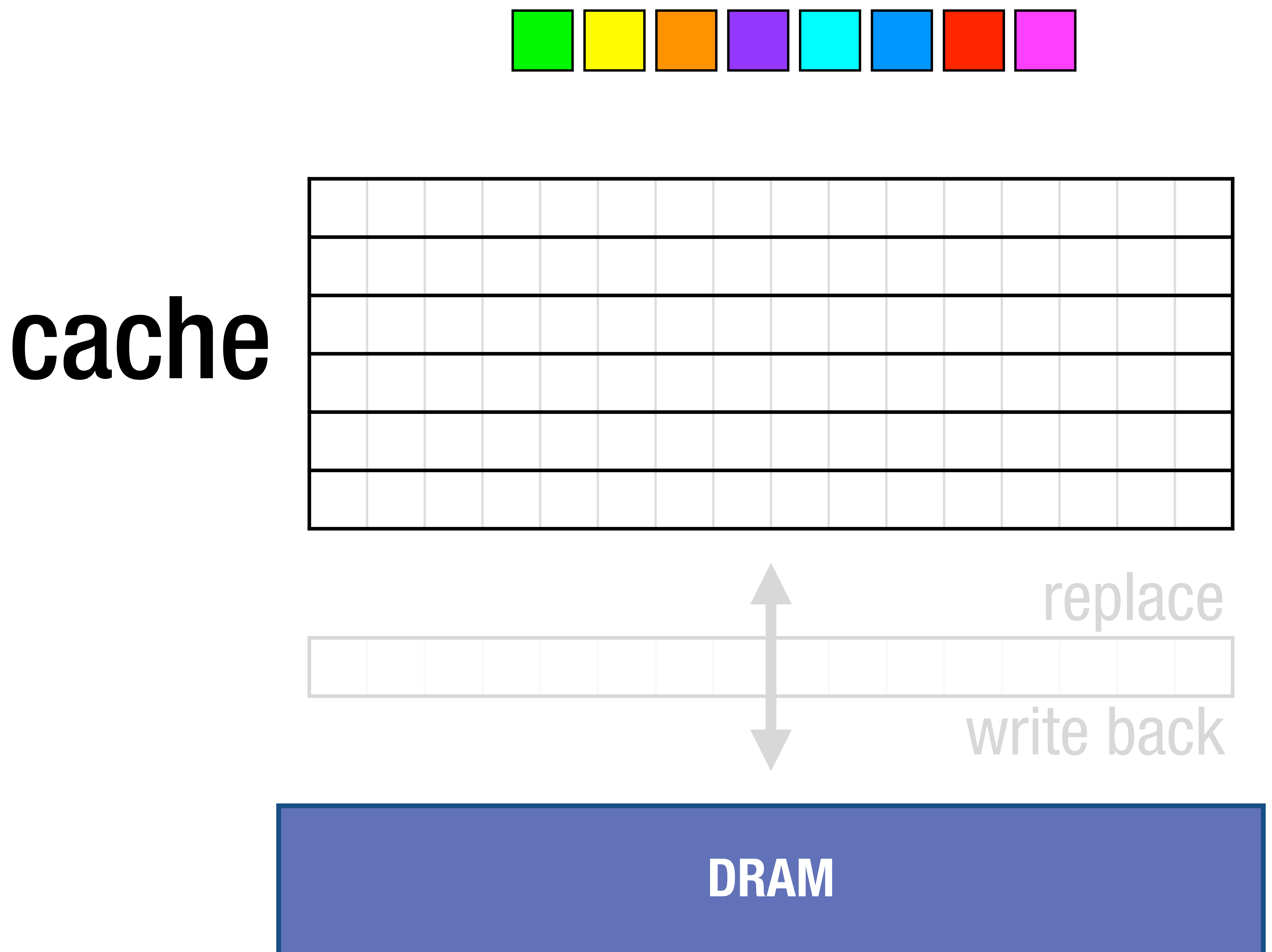



GPU Main Memory (20 GB)
L2 cache (48 MB)
L1 SRAM (6 MB)
Registers (12 MB)
EUs 6144

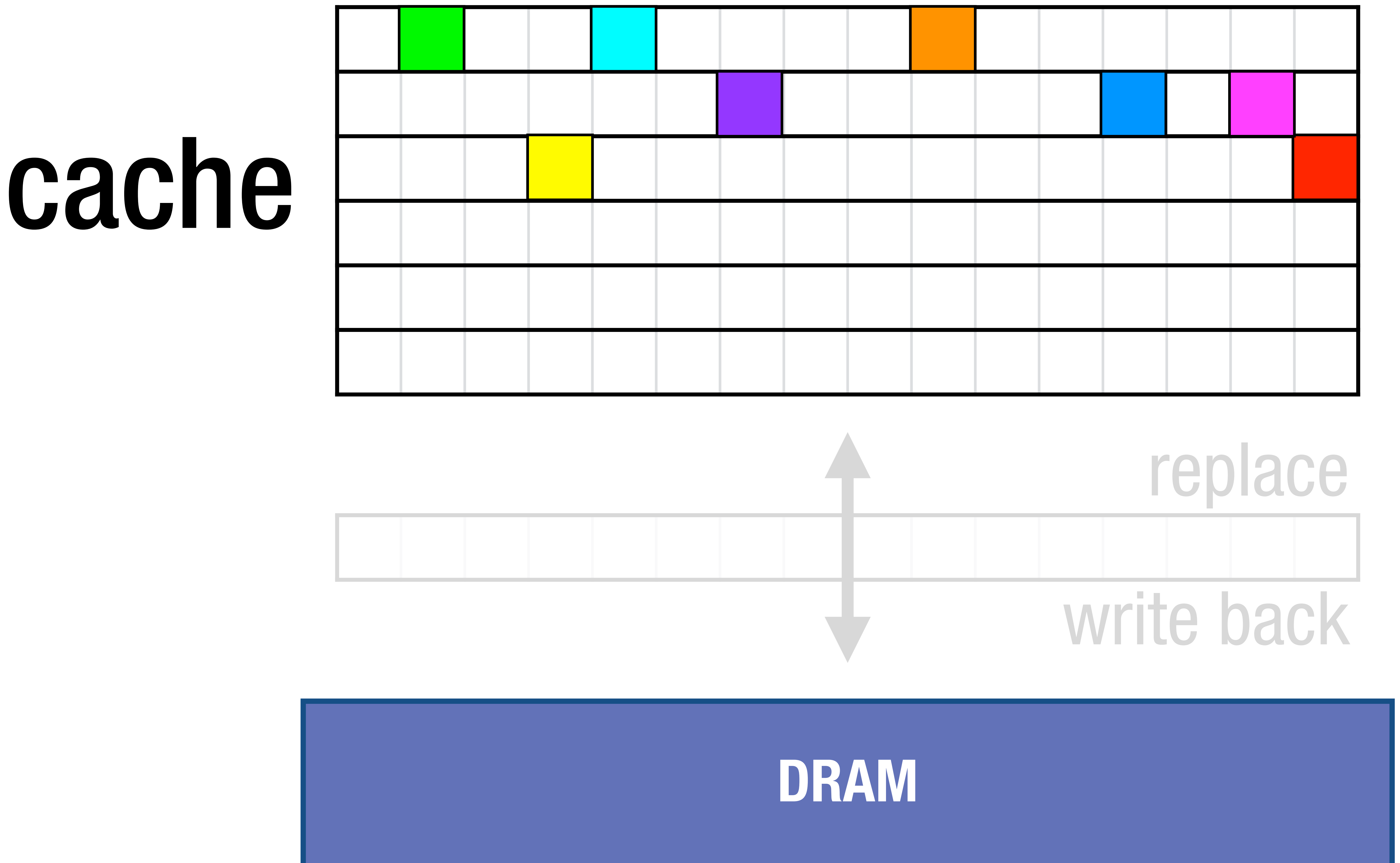
## **Approach 2:** cacheing coalesces mem. access



# **Approach 2:** cacheing coalesces mem. access

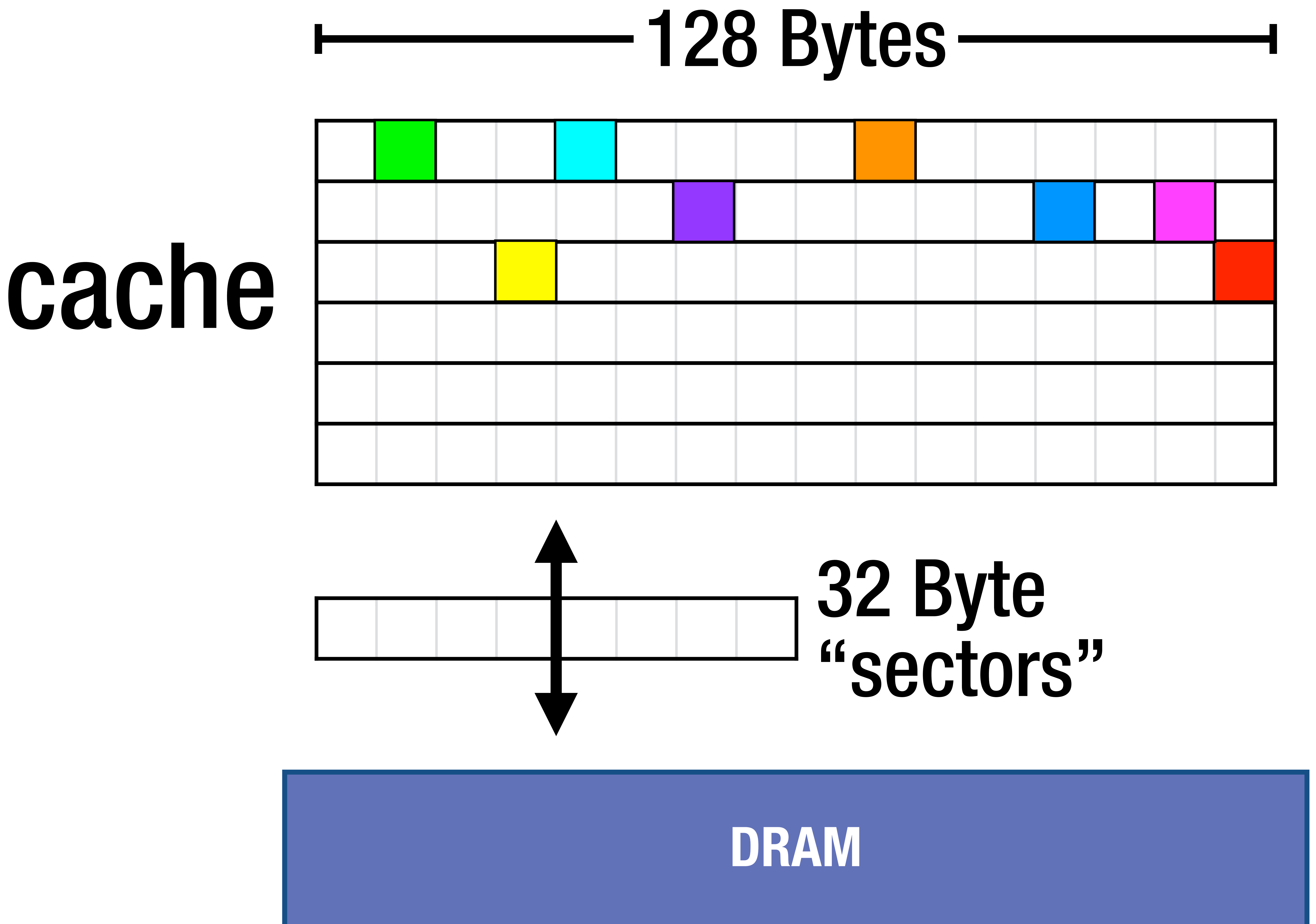


## **Approach 2:** cacheing coalesces mem. access



Block replacement **amortized** over potentially many accesses to the same line while cached.

# **Approach 2:** cacheing coalesces mem. access



Block replacement **amortized** over potentially many accesses to the same line while cached.

## Rule of thumb:

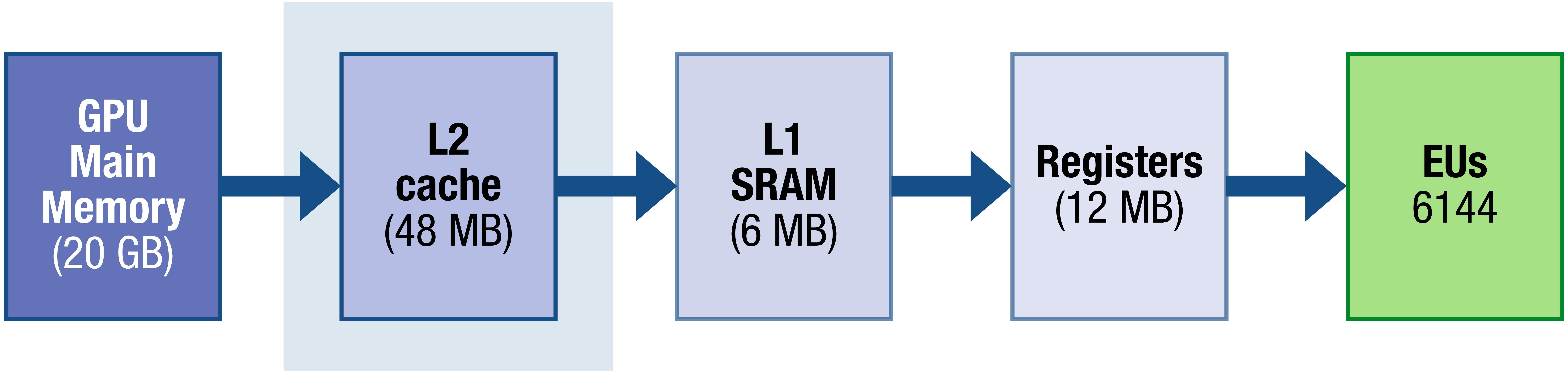
we can often **idealize** GPUs
in terms of **aggregate throughputs**

1 load /
150 ops

360
GB/sec

1 load /
20 ops

2.5
TB/sec



high **bandwidth**,
limited **capacity**

high clocks &
wide interface

**aggregate** large
transactions for
DRAM

**streaming**
access

large-scale **reuse**

1 load / 150 ops

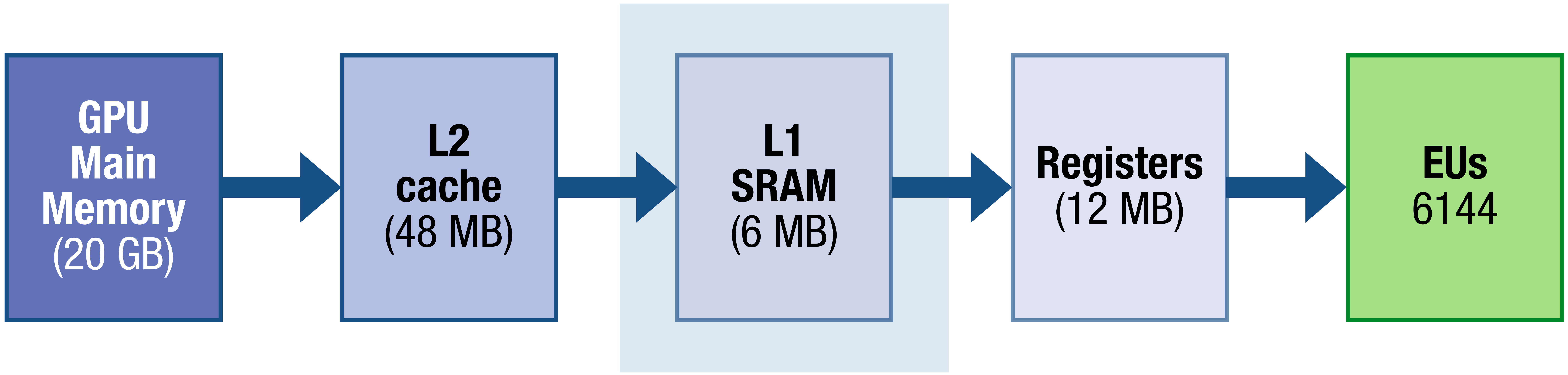
360 GB/sec

1 load / 20 ops

2.5 TB/sec

1 load / 4 ops

13.4 TB/sec



high **bandwidth**, limited **capacity**

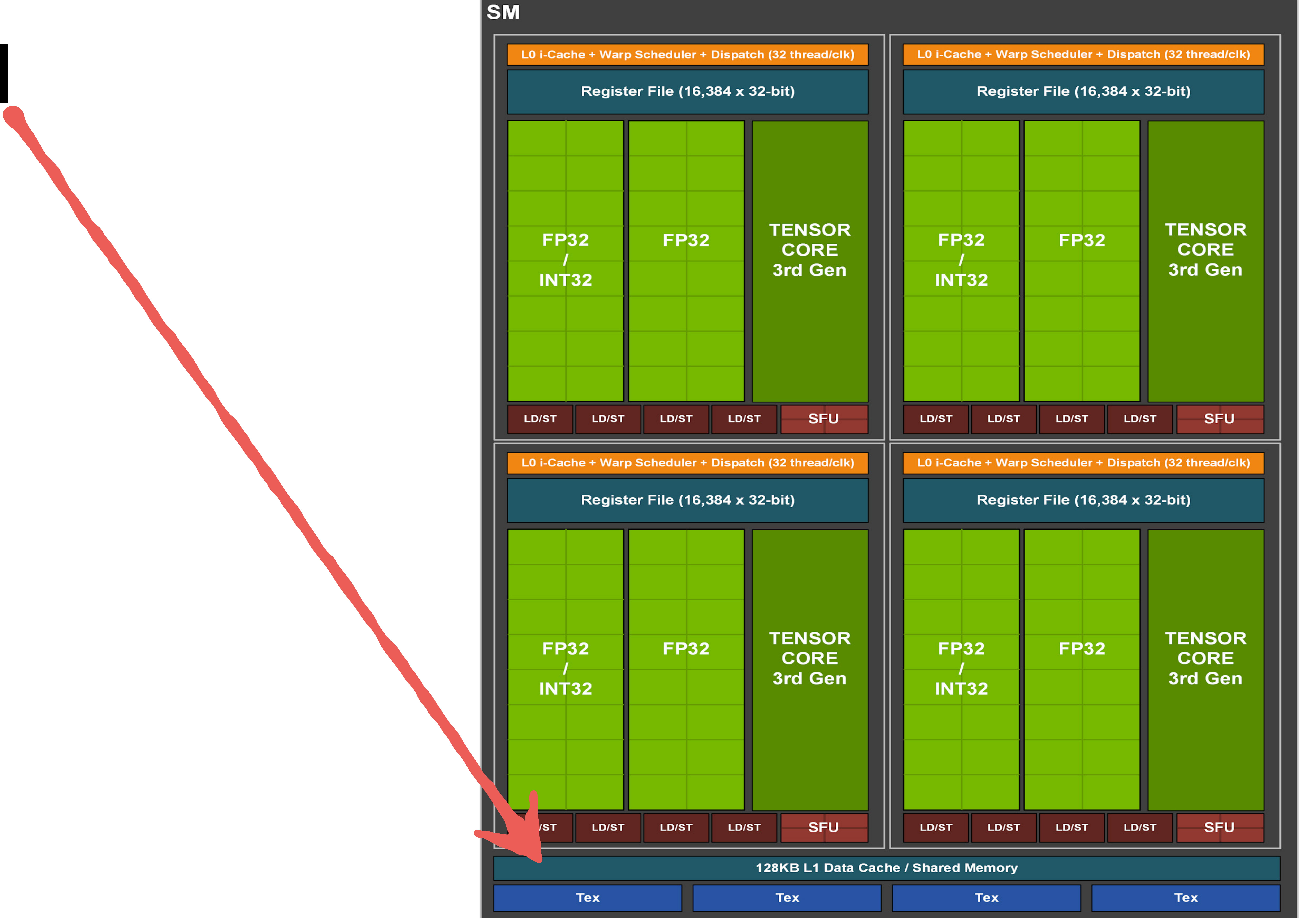
high clocks & wide interface

**aggregate** large transactions for DRAM

**streaming** access

large-scale **reuse**

# L1 SRAM


SM
L0 i-Cache + Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
FP32 / INT32
FP32
TENSOR CORE 3rd Gen
LD/ST
SFU
128KB L1 Data Cache / Shared Memory
Tex

## L1 SRAM

128 KB per-SM

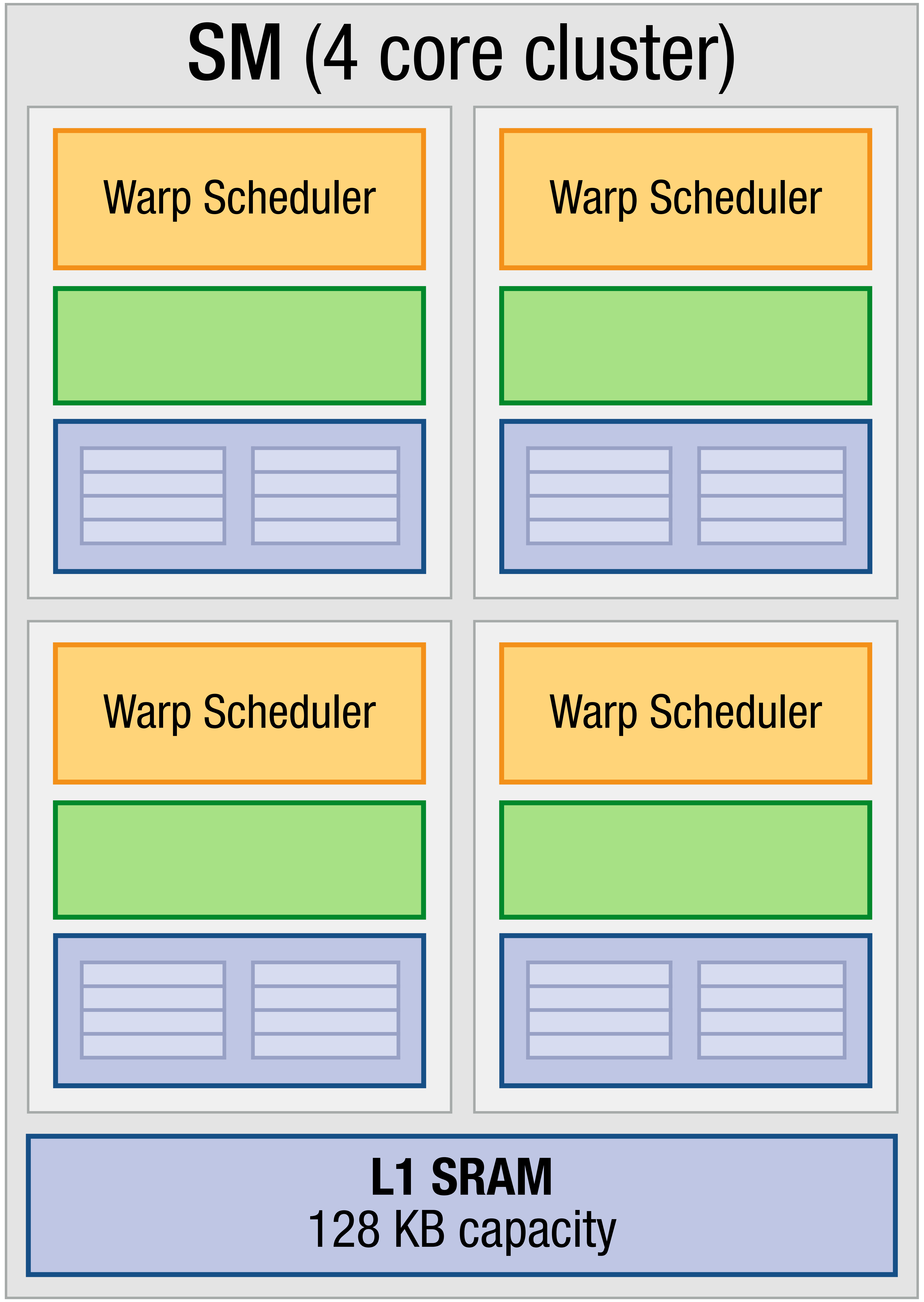
( $\times$ 48 SMs = 6 MB )

128 bytes / cycle / SM

$\hookrightarrow$ 1 warp-wide ld/st

(Per-core: 1 every 4 cycles)

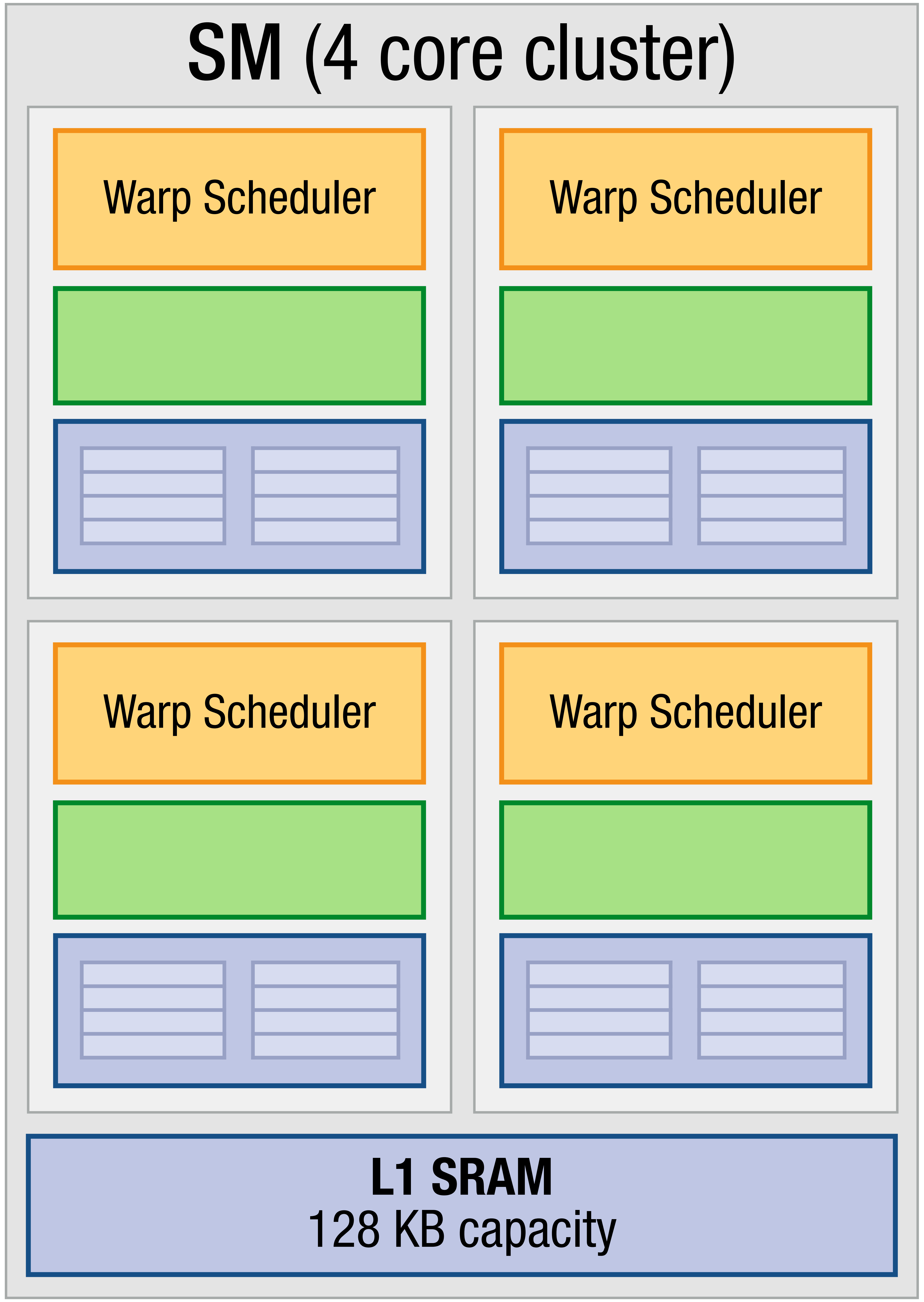
$\times$ 48 SMs $\times$ 2.18 GHz = 9.6 TB/s



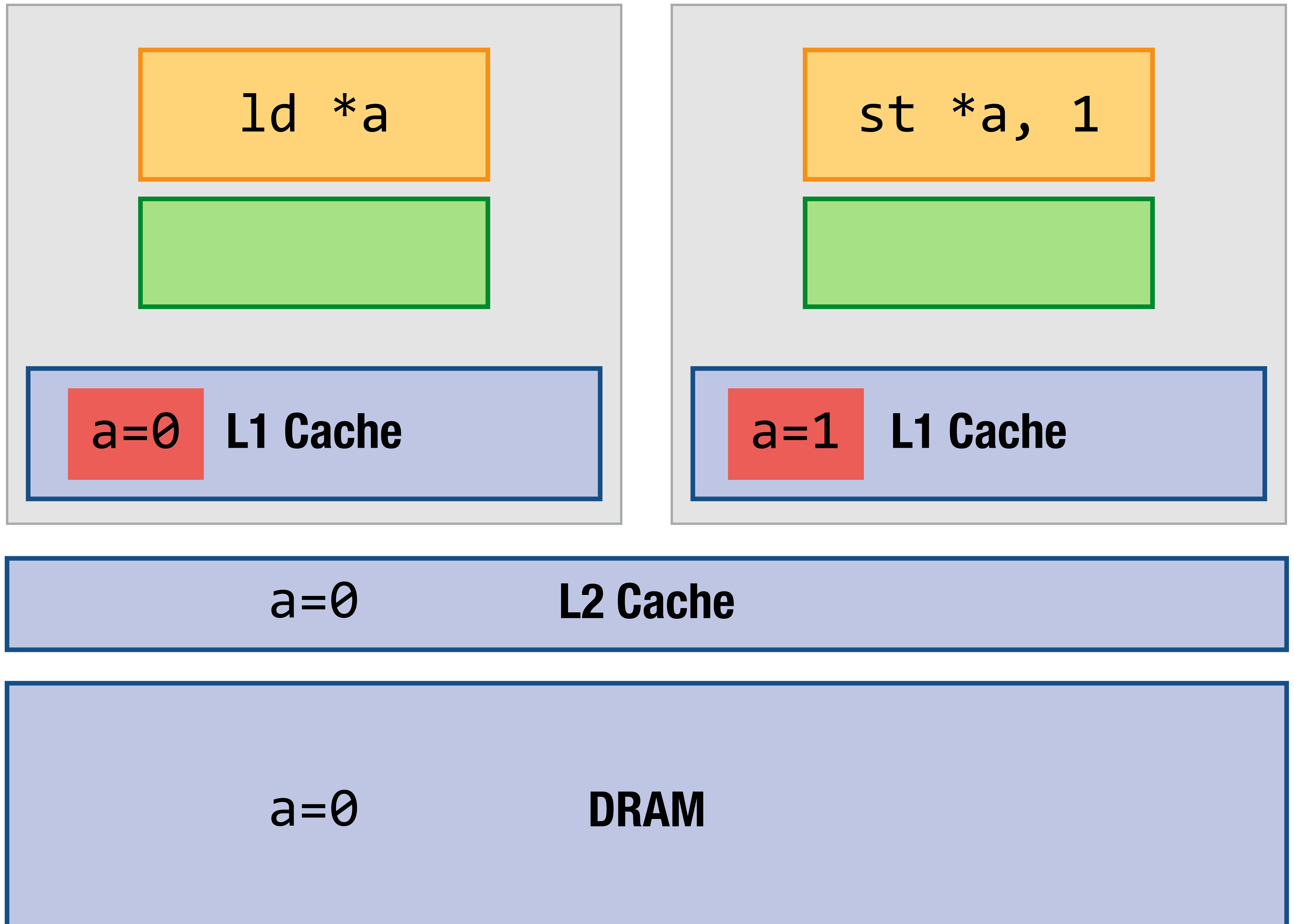
# L1 SRAM

Allows **data sharing & communication** across warps running **simultaneously** on the **same SM**.

In **CUDA**, grouping warps to allow this sharing is the role of **thread blocks**.



# **L1 Cache** is **incoherent** between SMs



## **L1 Cache** is **incoherent** between SMs

**Conventional processors** enforce **cache coherence** via complex protocols built into the hardware.

**Accelerators** often **forego coherence** in exchange for performance & scalability, at the cost of **programming complexity**.

## **L1 Cache:** opt-in via explicit instructions

`ld.local`
`st.local`

`ld.global.ca` vs.
`ld.global.cg`

**From CUDA C:**

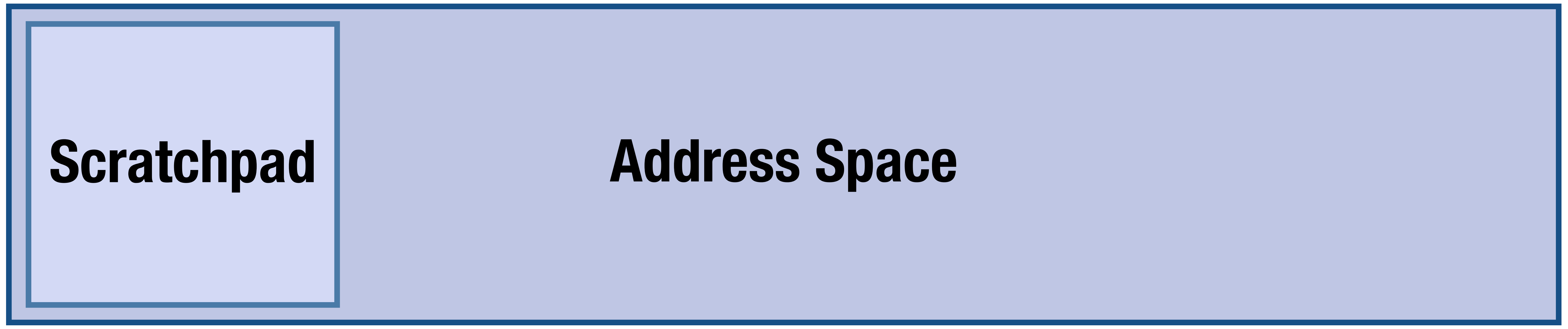
Read-only data
`const __restrict__`

`__ldg( )` intrinsic

`__local__`

Textures

## **L1 SRAM:** also used as **explicit scratchpad**



Each block (SM) only sees
its own scratchpad